Strong Collapse for Persistence

Jean-Daniel Boissonnat, Siddharth Pritam, Divyansh Pareek

Strong Collapse for Persistence*

Jean-Daniel Boissonnat[1], Siddharth Pritam[2], and Divyansh Pareek[3]

1 **Université Côte d'Azur, INRIA, France**
Jean-Daniel.Boissonnat@inria.fr
2 **Université Côte d'Azur, INRIA, France**
siddharth.pritam@inria.fr
3 **Université Côte d'Azur, INRIA, France**
divyanshpareek21@gmail.com

Abstract

We introduce a fast and memory efficient approach to compute the persistent homology (PH) of a sequence of simplicial complexes. The basic idea is to simplify the complexes of the input sequence by using strong collapses, as introduced by J. Barmak and E. Miniam [1], and to compute the PH of an induced sequence of reduced simplicial complexes that has the same PH as the initial one. Our approach has several salient features that distinguishes it from previous work. It is not limited to filtrations (i.e. sequences of nested simplicial subcomplexes) but works for other types of sequences like towers and zigzags. To strong collapse a simplicial complex, we only need to store the maximal simplices of the complex, not the full set of all its simplices, which saves a lot of space and time. Moreover, the complexes in the sequence can be strong collapsed independently and in parallel. As a result and as demonstrated by numerous experiments on publicly available data sets, our approach is extremely fast and memory efficient in practice.

1998 ACM Subject Classification Dummy classification – please refer to `http://www.acm.org/about/class/ccs98-html`

Keywords and phrases Computational Topology, Topological Data Analysis, Strong Collapse, Persistent homology

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

In this article, we address the problem of computing the Persistent Homology (PH) of a given sequence of simplicial complexes (defined precisely in Section 4) in an efficient way. It is known that computing persistence can be done in $O(n^\omega)$ time, where n is the total number of simplices and $\omega \leq 2.4$ is the matrix multiplication exponent [25, 17]. In practice, when dealing with massive and high-dimensional datasets, n can be very large (of order of billions) and computing PH is then very slow and memory intensive. Improving the performance of PH computation has therefore become an important research topic in Computational Topology and Topological Data Analysis.

Much progress has been accomplished in the recent years in two directions. First, a number of clever implementations and optimizations have led to a new generation of software for PH computation [37, 39, 40, 41]. Secondly, a complementary direction has been explored

* This research has received funding from the European Research Council (ERC) under the European Union's Seventh Framework Programme (FP/2007- 2013) / ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions).

to reduce the size of the complexes in the sequence while preserving (or approximating in a controlled way) the persistent homology of the sequence. Examples are the work of Mischaikow and Nanda [8] who use Morse theory to reduce the size of a filtration, and the work of Dłotko and Wagner who use simple collapses [13]. Both methods compute the exact PH of the input sequence. Approximations can also be computed with theoretical guarantees. Approaches like interleaving smaller and easily computable simplicial complexes, and sub-sampling of the point sample works well upto certain approximation factor [21, 12, 10, 9].

In this paper, we introduce a new approach to simplify the complexes of the input sequence which uses the notion of strong collapse introduced by J. Barmak and E. Miniam [1]. Specifically, our approach can be summarized as follows: Given a sequence $\mathcal{Z} : \{K_1 \xrightarrow{f_1} K_2 \xleftarrow{g_2} K_3 \xrightarrow{f_3} \cdots \xrightarrow{f_{(n-1)}} K_n\}$ of simplicial complexes K_i connected through simplicial maps $\{\xrightarrow{f_i}$ or $\xleftarrow{g_j}\}$. We independently strong collapse the complexes of the sequence to reach a sequence $\mathcal{Z}^c : \{K_1^c \xrightarrow{f_1^c} K_2^c \xleftarrow{g_2^c} K_3^c \xrightarrow{f_3^c} \cdots \xrightarrow{f_{(n-1)}^c} K_n^c\}$, with *induced simplicial maps* $\{\xrightarrow{f_i^c}$ or $\xleftarrow{g_j^c}\}$ (defined in Section 4). The complex K_i^c is called the **core** of the complex K_i and we call the sequence $\mathcal{Z}^c$ the **core sequence** of $\mathcal{Z}$. We show that one can compute the PH of the sequence $\mathcal{Z}$ by computing the PH of the core sequence $\mathcal{Z}^c$, which is of much smaller size.

Our method has some similarity with the work of Wilkerson et. al. [5] who also use strong collapses to reduce PH computation but it differs in three essential aspects: it is not limited to filtrations (i.e. sequences of nested simplicial subcomplexes) but works for other types of sequences like towers and zigzags. It also differs in the way the strong collapses are computed and in the manner PH is computed.

A first central observation is that to strong collapse a simplicial complex K, we only need to store its maximal simplices (i.e. those simplices that have no coface). The number of maximal simplices is smaller than the total number of simplices by a factor that is exponential in the dimension of the complex. It is linear in the number of vertices for a variety of complexes [32]. Working only with maximal simplices dramatically reduces the time and space complexities compared to the algorithm of [4]. We prove that the complexity of our algorithm is $\mathcal{O}(v^2\Gamma_0 d + m^2\Gamma_0 d)$. Here d is the dimension of the complex, v is the number of vertices, m is the number of maximal simplices and Γ_0 is an upper bound on the number of maximal simplices incident to a vertex. Γ_0 is a small fraction of the number of maximal simplices [31].

We now consider PH computation. All PH algorithms take as input a full representation of the complexes and their complexity is polynomial in the total number of simplices of the complexes. We thus have to convert the representation by maximal simplices used for the strong collapses into a full representation of the complexes, which takes exponential time in the dimension (of the collapsed complexes). This exponential burden is to be expected since it is known that computing PH is NP-hard when the complexes are represented by their maximal faces [7]. Nevertheless, we demonstrate in this paper that strong collapses combined with known persistence algorithms lead to major improvements over previous methods to compute the PH of a sequence. This is due in part to the fact that strong collapses reduce the size of the complexes on which persistence is computed. The following factors also contribute:

– The collapses of the complexes in the sequence can be performed independently and in parallel. This is due to the fact that strong collapses can be expressed as simplicial maps unlike simple collapses [30].

– The size of the complexes in a sequence does not grow by much in terms of maximal simplices, as observed in many practical cases. As a consequence, the time to collapse the i-th simplicial complex K_i in the sequence is almost independent of i. For filtrations, this is

a clear advantage over methods that use a full representation of the complexes and suffer an increasing cost as i increases.

As a result, our approach is extremely fast and memory efficient in practice as demonstrated by numerous experiments on publicly available data sets.

An outline of this paper is as follows. Section 2 recalls the basic ideas and constructions related to simplicial complexes and strong collapses. We describe our core algorithm in Section 3. In Section 4, we prove that zigzag modules are preserved under strong collapse. In Section 5, we provide experimental results.

2 Preliminaries

In this section, we provide a brief review of the notions of simplicial complex and strong collapse as introduced in [1]. We assume some familiarity with basic concepts like homotopic maps, homotopy type, homology groups and other algebraic topological notions. Readers can refer to [29] for a comprehensive introduction of these topics.

Simplex, simplicial complex and simplicial map : An **abstract simplicial complex** K is a collection of subsets of a non-empty finite set X, such that for every subset A in K, all the subsets of A are in K. From now on we will call an *abstract simplicial complex* simply a *simplicial complex* or just a *complex*. An element of K is called a **simplex**. An element of cardinality $k+1$ is called a k-simplex and k is called its **dimension**. A simplex is called **maximal** if it is not a proper subset of any other simplex in K. A sub-collection L of K is called a **subcomplex**, if it is a simplicial complex itself. L is a **full subcomplex** if it contains all the simplices of K that are spanned by the vertices (0-simplices) of the subcomplex L.

A vertex to vertex map $\psi : K \to L$ between two simplicial complexes is called a **simplicial map**, if the images of the vertices of a simplex always span a simplex. Simplicial maps are thus determined by the images of the vertices. In particular, there are a finite number of simplicial maps between two given finite simplicial complexes. Simplicial maps induce continuous maps between the underlying *geometric realisations* of the simplicial complexes.

Dominated vertex: Let σ be a simplex of a simplicial complex K, the **closed star** of σ in K, $st_K(\sigma)$ is a subcomplex of K which is defined as follows, $st_K(\sigma) := \{\tau \in K |\ \tau \cup \gamma \in K;\ \textit{where}\ \gamma \subseteq \sigma\}$. The **link** of σ in K, $lk_K(\sigma)$ is defined as the set of simplices in $st_K(\sigma)$ which do not intersect with σ, $lk_K(\sigma) := \{\tau \in st_K(\sigma) | \tau \cap \sigma = \emptyset\}$.

Taking a join with a vertex transforms a simplicial complex into a simplicial cone. Formally if L is a simplicial complex and a is vertex not in L then the **simplicial cone** aL is defined as, $aL := \{a, \tau |\ \tau \in L\ \textit{or}\ \tau = \sigma \cup a;\ \textit{where}\ \sigma \in L\}$. A vertex v in K is called a **dominated vertex** if the link of v in K, $lk_K(v)$ is a simplicial cone, that is, there exists a vertex $v' \neq v$ and a subcomplex L in K, such that $lk_K(v) = v'L$. We say that the vertex v' is *dominating* v and v is *dominated* by v'. The symbol $K \setminus v$ (deletion of v from K) corresponds to the subcomplex of K which has all simplices of K except the ones containing v. Below is an important remark from [1, Remark 2.2], which proposes an alternative definition of dominated vertices.

Remark 1: A vertex $v \in K$ is dominated by another vertex $v' \in K$, *iff* all the maximal simplices of K that contain v also contain v' [1].

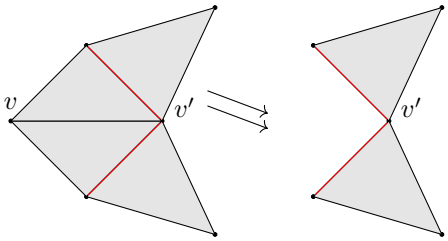


Figure 1 Illustration of an *elementary strong collapse*. In the complex on the left, v is dominated by v'. The link of v is highlighted in red. Removing v leads to the complex on the right.

Strong collapse: An **elementary strong collapse** is the deletion of a dominated vertex v from K, which we denote with $K \searrow\!\searrow^{e} K \setminus v$. Figure 1 illustrates an easy case of an elementary strong collapse. There is a **strong collapse** from a simplicial complex K to its subcomplex L, if there exists a series of elementary strong collapses from K to L, denoted as $K \searrow\!\searrow L$. The inverse of a strong collapse is called a **strong expansion**. If there exists a combination of strong collapses and/or strong expansion from K to L then K and L are said to have the same **strong homotopy type**.

The notion of strong homotopy type is stronger than the notion of simple homotopy type in the sense that if K and L have the same strong homotopy type, then they have the same simple homotopy type, and therefore the same homotopy type [1]. There are examples of contractible or simply collapsible simplicial complexes that are not strong collapsible.

A complex without any dominated vertex will be called a **minimal complex**. A **core** of a complex K is a minimal subcomplex $K^c \subseteq K$, such that $K \searrow\!\searrow K^c$. *Every simplicial complex has a* ***unique core*** *up to isomorphism. The core decides the strong homotopy type of the complex*, and two simplicial complexes have the same strong homotopy type *iff* they have isomorphic cores [1, Theorem 2.11].

Retraction map: If v' dominates v in K. The vertex map $r : K \to K \setminus v$ defined as: $r(w) = w$ if $w \neq v$ and $r(v) = v'$, induces a simplical map that is a *retraction* map. The homotopy between r and the identity over $K \setminus v$ is in fact a strong deformation retract.

Nerve of a simplicial complex: An open **cover** $\mathcal{U}$ of a topological space $\mathcal{X}$ is a set of open sets of $\mathcal{X}$, such that $\mathcal{X}$ is a subset of their union. The **nerve** of a cover $\mathcal{U}$ is an abstract simplicial complex, defined as the set of all non-empty intersections of the elements of $\mathcal{U}$. The nerve is a well known construction that transforms a continuous space to a combinatorial space preserving its homotopy type. The *nerve* $\mathcal{N}(K)$ of a simplicial complex K is defined as the nerve of the set of maximal simplices of the complex K (considered as a cover of the complex). Hence all the maximal simplices of K will be the vertices of $\mathcal{N}(K)$ and their non-empty intersection will form the simplices of $\mathcal{N}(K)$. For $j \geq 2$ the iterative construction is defined as $\mathcal{N}^j(K) = \mathcal{N}(\mathcal{N}^{j-1}(K))$. This definition of nerve preserves the homotopy type, $K \simeq \mathcal{N}(K)$[1]. A remarkable property of this nerve construction is its connection with strong collapses.

Taking the nerve of any simplicial complex K twice corresponds to a strong collapse. We will state it as a theorem below and recall the proof from [1] in the Appendix.

▶ **Theorem 1.** *[1, Proposition 3.4] For a simplicial complex K, there exists a subcomplex L isomorphic to $\mathcal{N}^2(K)$, such that $K \searrow\!\searrow L$.*

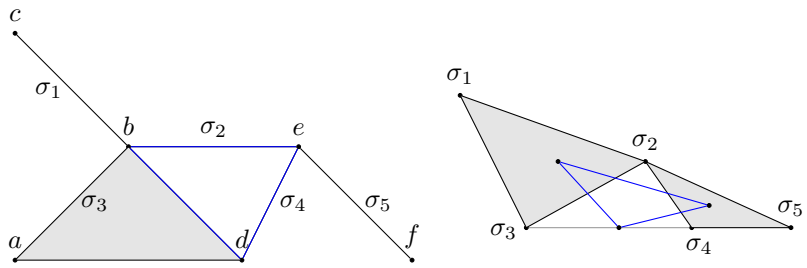


Figure 2 Left: K (in grey), Right: $\mathcal{N}(K)$ (in grey) and $\mathcal{N}^2(K)$ (in blue). $\mathcal{N}^2(K)$ is isomorphic to a full-subcomplex of K highlighted in blue on the left.

An easy consequence of this theorem is that a complex K is *minimal* if and only if it is isomorphic to $\mathcal{N}^2(K)$ [1, Lemma 3.6]. This means that we can keep collapsing our complex K by applying $\mathcal{N}^2(-)$ iteratively until we reach the core of the complex K. The sequence $K, \mathcal{N}^2(K), ..., \mathcal{N}^{2p}(K)$ is a decreasing sequence up to isomorphism. It leads to the easy however deep consequence that K is strong collapsible *iff* $\mathcal{N}^{2p}(K)$ is a point for some $p \geq 1$ [1].

3 Algorithm

In this section, we describe an algorithm to strong collapse a simplicial complex K, provide the details of the implementation and analyze its complexity. We construct $\mathcal{N}^2(K)$ as defined in Section 2.

Data structure: Basically, we represent K as the adjacency matrix M between the vertices and the maximal simplices of K. We will simply call M the adjacency matrix of K. The rows of M represent the vertices and the columns represent the maximal simplices of K. For convenience, we will identify a row (resp. column) and the vertex (resp. maximal simplex) it represents. An entry $M[v_i][\sigma_j]$ associated to a vertex v_i and a maximal simplex σ_j is set to 1 if $v_i \in \sigma_j$, and to 0 otherwise. For example, the matrix M in the left of the Table 1 corresponds to the leftmost simplicial complex K in Figure 2. Usually, M is very sparse. Indeed, each column contains at most $d+1$ non-zero element since the simplices of a d-dimensional complex have at most $d+1$ vertices, and each line contains at most Γ_0 non-zero elements where Γ_0 is an upper bound on the number of maximal simplices incident to a given vertex. In most practical situations, Γ_0 is a small fraction of the number of maximal simplices [31, Table 5]. It is therefore beneficial to store M as a list of vertices, each vertex giving access to a list of maximal simplices that contain the vertex, and a list of maximal simplices, each simplex giving access to its list of vertices. This is similar to the SAL data structure of [31].

Core algorithm: Given the adjacency matrix M of K, we compute the adjacency matrix C of the *core* K^c. It turns out that using basic row and column removal operations, we can easily compute C from M. Loosely speaking our algorithm recursively computes $\mathcal{N}^2(K)$ until it reaches K^c.

The columns of M (which represent the maximal simplices of K) correspond to the vertices of $\mathcal{N}(K)$. Also, the columns of M that have a non-zero value in a particular row v correspond to the maximal simplices of K that share the vertex associated to row v.

	σ_1	σ_2	σ_3	σ_4	σ_5
a	0	0	1	0	0
b	1	1	1	0	0
c	1	0	0	0	0
d	0	0	1	1	0
e	0	1	0	1	1
f	0	0	0	0	1

	b	d	e
σ_1	1	0	0
σ_2	1	0	1
σ_3	1	1	0
σ_4	0	1	1
σ_5	0	0	1

	σ_2	σ_3	σ_4
b	1	1	0
d	0	1	1
e	1	0	1

Table 1 From left to right M, $\mathcal{N}(M)$ and $\mathcal{N}^2(M)$

Therefore, each row of M represents a simplex of the nerve $\mathcal{N}(K)$. Not all the simplices of $\mathcal{N}(K)$ are associated to rows of M but all maximal simplices are since they correspond to subsets of maximal simplices with a common vertex. To remedy this situation, we *remove* all the rows of M that correspond to non-maximal simplices of $\mathcal{N}(K)$. This results in a new smaller matrix M whose transpose, noted $\mathcal{N}(M)$, is the adjacency matrix of the nerve $\mathcal{N}(K)$. We then exchange the roles of rows and columns (which is the same as taking the transpose) and run the very same procedure as before so as to obtain the adjacency matrix $\mathcal{N}^2(M)$ of $\mathcal{N}^2(K)$.

The process is iterated as long as the matrix can be reduced. Upon termination, we output the reduced matrix $C := \mathcal{N}^{2p}(M)$, for some $p \geq 1$, which is the adjacency matrix of the core K^c of K. Removing a row or column is the most basic operation of our algorithm. We will discuss it in more detail later in the paragraph *Domination test*.

Example: As mentioned before, the matrix M in the left of the Table 1 represents the simplicial complex K in the left of Figure 2. We go through the rows first, rows a and c are subsets of row b and row f is a subset of e. Removing rows a, c and f and transposing it, yields the adjacency matrix $\mathcal{N}(M)$ of $\mathcal{N}(K)$ in the middle. Now, row σ_1 is a subset of σ_2 and of σ_3, also σ_5 is a subset of σ_2 and of σ_4. We remove these two rows of $\mathcal{N}(M)$ and transpose it and get the rightmost matrix $\mathcal{N}^2(M)$, which corresponds to the core drawn in blue in Figure 2.

Domination test: Now we explain in more detail how to detect the rows that need to be removed. Let v be a row of M and σ_v be the associated simplex in $\mathcal{N}(K)$. If σ_v is not a maximal simplex of $\mathcal{N}(K)$, it is a proper face of some maximal simplex $\sigma_{v'}$ of $\mathcal{N}(K)$. Equivalently, the row v' of M that is associated to $\sigma_{v'}$ contains row v in the sense that the non zero elements of v appear in the same columns as the non zero elements of v'. We will say that row v is dominated by row v' and determining if a row is dominated by another one will be called the row domination test. Notice that when a row v is dominated by a row v', the same is true for the associated vertices since all the maximal simplices that contain vertex v also contain vertex v', which is the criterion to determine if v is dominated by v' (See Remark 1 in Section 2). The algorithm removes all dominated rows and therefore all dominated vertices of K.

After removing rows, the algorithm removes the columns that are no longer maximal in K, which might happen since we removed some rows. Removing a column may lead in turn to new dominated vertices and therefore new rows to be removed. When the algorithms stops, there are no rows to be removed and we have obtained the core K^c of the complex K.

The algorithm in fact provides a constructive proof of Theorem 1.

Removing columns is done in very much the same way: we just exchange the roles of rows and columns.

Computing the retraction map r: The algorithm also provides a direct way to compute the retraction map r defined in Section 2. The retraction map corresponding to the strong collapses executed by the algorithm can be constructed as follows. A row being removed in M corresponds to a dominated vertex in K and the row which *contains* it corresponds to a dominating vertex. Therefore we map the dominated vertex to the dominating vertex and compose all such maps to get to the final retraction map from K to its core K^c. The final map is simplicial as well as it is a composition of simplicial maps.

Reducing the number of domination tests: We first observe that, when one wants to determine if a row v is dominated by some other row, we don't need to test v with all other rows but with at most d of them. Indeed, at most $d+1$ rows can intersect a given column since a simplex can have at most $d+1$ vertices. For example, in Table 1 (Left), to check if row e (highlighted in brown) is dominated by another row, we pick the first non-zero column σ_2 (highlighted in Gray) and compare e with the non-zero entries {b} of σ_2.

A second observation is that we don't need to test all rows and columns for domination, but only the so-called candidate rows and columns. We define a row to be a **candidate row** for the next iteration if at least one column containing one of its non-zero elements has been removed in the previous column removal iteration. Similarly, by exchanging the roles of rows and columns, we define the **candidate columns**. Candidate rows and columns are the only rows or columns that need to be considered in the *domination* tests of the algorithm. Indeed, a column τ of M whose non-zero elements all belong to rows that are present from the previous *iteration* cannot be dominated by another column τ' of M, since τ was not dominated at the previous iteration and no new non-zero elements have been added. The same argument follows for the candidate rows.

We maintain two *queues*, one for the candidate columns (colQueue) and one for the candidate rows (rowQueue). These queues are implemented as First in First out (FIFO) queues. At each iteration, we *pop out* a candidate row or column from its respective queue and test whether it is dominated or not. After each successful domination test, we *push* the candidate columns or rows in their appropriate queue in preparation for the subsequent iteration. In the first iteration, we *push* all the rows in rowQueue and then alternatively use colQueue and rowQueue. Algorithm 1 gives the pseudo code of our algorithm.

Time Complexity: The most basic operation in our algorithm is to determine if a row is a dominated by another given row, and similarly for columns. In our implementation, the rows (columns) of the matrix that are considered by the algorithm are stored as sorted lists. Checking if one sorted list is a subset of another sorted list can be done in time $\mathcal{O}(l)$, where l is the size of the longer list. Note that the length of a row list is at most Γ_0 where Γ_0 denotes an upper bound on the number of maximal simplices incident to a vertex. The length of a column list is at most $d+1$ where d is the dimension of the complex. Hence checking if a row is dominated by another row takes $\mathcal{O}(\Gamma_0)$ time and checking if a column is dominated by another column takes $\mathcal{O}(d)$ time.

At each iteration on the rows (Lines 7-13 of Algorithm 1), each row is checked against at most d other rows (since a maximal simplex has at most $d+1$ vertices), and at each iteration of the columns (Lines 18-24 of Algorithm 1), each column is checked against at most Γ_0

Algorithm 1 Core algorithm

```
1: procedure CORE(M)                ▷ Returns the matrix corresponding to the core of K
2:     rowQueue ← push all rows of M (all vertices of K)
3:     colQueue ← empty
4:     while rowQueue is not empty do
5:         v ← pop(rowQueue)
6:         σ ← the first non-zero column of v
7:         for non-zero rows w in σ do
8:             if v is a subset of w then
9:                 Remove v from M
10:                push all non-zero columns τ of v to colQueue if not pushed before
11:                break
12:            end if
13:        end for
14:    end while
15:    while colQueue is not empty do
16:        τ ← pop(colQueue)
17:        v ← the first non-zero row of τ
18:        for non-zero columns σ in v do
19:            if τ is subset of σ then
20:                Remove τ from M
21:                push all non-zero rows w of τ to rowQueue if not pushed before
22:                break
23:            end if
24:        end for
25:    end while
26:    if rowQueue is not empty then
27:        GOTO 4
28:    end if
29:    return M                     ▷ The core consists of the remaining rows and columns
30: end procedure
```

other columns (since a vertex can belong to at most Γ_0 maximal simplices). Since, at each iteration on the rows, we remove at least one row, the total number of iterations on the rows is at most $O(v^2)$, where v is the total number of vertices of the complex K. Similarly, at each iteration on the columns, we remove at least one column and the total number of iterations on columns is $O(m^2)$, where m is the total number of maximal simplices of the complex K. The worst-case time complexity of our algorithm is therefore $\mathcal{O}(v^2\Gamma_0 d + m^2\Gamma_0 d)$. Observe that m is exponentially smaller (wrt d) than the total number of simplices. Moreover $\Gamma_0 \leq m/n$ and is usually a small fraction of the number of maximal simplices [31].

4 Strong collapse of zigzag sequences

In this section, we begin with some brief background on quiver theory and zigzag persistence, enough to present our main result. Readers interested in more details can refer to [22, 23, 26].

Quivers: A **quiver** $\mathcal{Q}$ is a directed graph, more formally it's a pair of sets (Q_0, Q_1), where $Q_0 = \{v_1, v_2, ...v_n\}$; $n \in \mathbb{N}$ is the set of vertices and Q_1 is the set of directed edges between the vertices. Given a field $\mathbb{F}$, an **$\mathbb{F}$-representation** $\mathbb{V} = (V_i, f_{ij})$ of a quiver $\mathcal{Q}$ is an assignment of a finite dimensional $\mathbb{F}$-vector spaces V_i for every vertex $v_i \in Q_0$ and a homomorphism $f_{ij} : V_i \to V_j$ for each directed edge $e_{ij} \in Q_1$. A **morphism** ϕ between two different $\mathbb{F}$-representations $\mathbb{V} = (V_i, f_{ij})$ and $\mathbb{W} = (W_i, g_{ij})$ of a quiver $\mathcal{Q}$ is a set of homomorphisms $\phi_i : V_i \to W_i$ such that, for every edge $e_{ij} \in Q_1$, $\phi_j f_{ij} = g_{ij}\phi_i$. In other words all the squares in two representations connected by ϕ_is commute. A morphism ϕ is an **isomorphism** ($\cong$) if all individual ϕ_is are bijective. A **direct sum** $\mathbb{V} \oplus \mathbb{W}$ between two different $\mathbb{F}$-representations is defined using the direct sum of the individual vector spaces $V_i \oplus W_i$ at each vertex v_i and direct sum (component-wise application) homomorphisms $f_{ij} \oplus g_{ij}$ for each edge e_{ij}. An $\mathbb{F}$-representation $\mathbb{V}$ is **decomposable** if it is isomorphic to a direct sum of two non-trivial $\mathbb{F}$-representations. Otherwise it is called **indecomposable**. These indecomposable $\mathbb{F}$-representations of a quiver can be thought of as building blocks of the decomposable representations.

Zigzag module: A **zigzag module** is a "topological" representation of a quiver whose underlying graph is a Dynkin graph of type A_n. Here the vector spaces are (co)homology classes with induced homomorphisms between them. If a quiver $\mathcal{Q}$ is of type A_n, for two integers b and d, $0 \leq b \leq d \leq n$; we can define an interval $\mathbb{F}$-representation $\mathbb{I}[b, d]$ of $\mathcal{Q}$ by assigning the vector space $\mathbb{F}$ for each v_i when $i \in [b, d]$, and null spaces otherwise, the maps between any two $\mathbb{F}$ vector space is identity and is zero otherwise. These intervals representations are precisely the indecomposable representations for quivers of type A_n. A result by Gabriel [22, Theorem 11] says that every $\mathbb{F}$-representation $\mathbb{V}$ of a quiver of type A_n can be decomposed as the direct sum of *finitely* many interval representations. In particular this means that a zigzag module can be represented with a finite set of intervals. These intervals pertain topological information as they indicate *birth* and *death* of homology classes. The multiset of all the intervals $[b_j, d_j]$ is called a **zigzag (persistence) diagram**. The zigzag diagram completely characterizes the zigzag module, that is, there is bijective correspondence between them [22, 18].

Given a **zigzag sequence** of simplicial complexes $\mathcal{Z} : \{K_1 \xrightarrow{f_1} K_2 \xleftarrow{g_2} K_3 \xrightarrow{f_3} \cdots \xrightarrow{f_{(n-1)}} K_n\}$, the most common way to get a zigzag module is to compute homology classes of these sequences with field coefficients. The zigzag module of $\mathcal{Z}$, $\mathcal{P}(\mathcal{Z}) : \{H_p(K_1) \xrightarrow{f_1^*} H_p(K_2) \xleftarrow{g_2^*}$

$H_p(K_3) \xrightarrow{f_3^*} \cdots \xrightarrow{f^*_{(n-1)}} H_p(K_n)\}$. Here p is the dimension of the homology class and $*$ denotes the induced homomorphisms. A zigzag sequence is called a simplicial **tower** if all maps are forward. i.e. only f_is. A tower is called **filtration** if the maps are only inclusions. A zigzag module arising from a sequence of simplicial complexes captures the evolution of the topology of the sequence, encoded it in its zigzag (persistence) diagram.

Strong collapse of the zigzag module: Given a zigzag sequence $\mathcal{Z} : \{K_1 \xrightarrow{f_1} K_2 \xleftarrow{g_2} K_3 \xrightarrow{f_3} \cdots \xrightarrow{f_{(n-1)}} K_n\}$. We define the **core sequence** $\mathcal{Z}^c$ of $\mathcal{Z}$ as $\mathcal{Z}^c : \{K_1^c \xrightarrow{f_1^c} K_2^c \xleftarrow{g_2^c} K_3^c \xrightarrow{f_3^c} \cdots \xrightarrow{f^c_{(n-1)}} K_n^c\}$. Where K_i^c is the core of K_i. The forward maps are defined as, $f_j^c := r_{j+1} f_j i_j$; and the backward maps are defined as $g_j^c := r_j g_j i_{j+1}$. The maps $i_j : K_j^c \hookrightarrow K_j$ and $r_j : K_j \to K_j^c$ are the composed inclusions and the retractions maps defined in Section 2 respectively.

▶ **Theorem 2.** *Zigzag modules $\mathcal{P}(\mathcal{Z})$ and $\mathcal{P}(\mathcal{Z}^c)$ are equivalent.*

Proof. Consider the following diagram

$$\begin{array}{ccccccccc}
K_1 & \xrightarrow{f_1} & K_2 & \xleftarrow{g_2} & K_3 & \cdots & \longrightarrow & K_{n-1} & \xrightarrow{f_{n-1}} & K_n \\
\downarrow{\scriptstyle r_1} & & \downarrow{\scriptstyle r_2} & & \downarrow{\scriptstyle r_3} & & & \downarrow{\scriptstyle r_{n-1}} & & \downarrow{\scriptstyle r_n} \\
K_1^c & \xrightarrow{f_1^c} & K_2^c & \xleftarrow[g_2^c]{} & K_3^c & \cdots & \longrightarrow & K_{n-1}^c & \xrightarrow{f_{n-1}^c} & K_n^c
\end{array}$$

and the associated diagram after computing the p-th homology groups

$$\begin{array}{ccccccccc}
H_p(K_1) & \xrightarrow{f_1^*} & H_p(K_2) & \xleftarrow[g_2^*]{} & H_p(K_3) & \cdots & \longrightarrow & H_p(K_{n-1}) & \xrightarrow{f_{n-1}^*} & H_p(K_n) \\
\downarrow{\scriptstyle r_1^*} & & \downarrow{\scriptstyle r_2^*} & & \downarrow{\scriptstyle r_3^*} & & & \downarrow{\scriptstyle r_{n-1}^*} & & \downarrow{\scriptstyle r_n^*} \\
H_p(K_1^c) & \xrightarrow{(f_1^c)^*} & H_p(K_2^c) & \xleftarrow[(g_2^c)^*]{} & H_p(K_3^c) & \cdots & \longrightarrow & H_p(K_{n-1}^c) & \xrightarrow{(f_{n-1}^c)^*} & H_p(K_n^c)
\end{array}$$

$*$ represents the induced homomorphisms between the homology classes by the corresponding simplicial maps. Since there exists a strong deformation retract between r_j and i_j, the induced homomorphisms r_j^* and i_j^* are isomorphisms [29, Corollary 2.11]. Also, $(f_j^c)^* = (r_{j+1} f_j i_j)^* = r_{j+1}^* f_j^* i_j^* = f_j^*$ and similarly $(g_j^c)^* = g_j^*$, see [29, Proposition (1) page 111]. Therefore all the squares in the lower diagram commute. Now the top and bottom rows are two different representations of the same underlying quiver. The set of maps r_j^*s are bijective (isomorphisms on the vector spaces), therefore these two representations are isomorphic and hence their zigzag diagrams are identical. ◀

In fact, using the notion of quiver representation this result follows for the multidimensional persistence as well.

5 Computational experiments

For each data set, we consider as the input sequence a nested sequence (filtration) of Vietoris-Rips (VR) complexes associated to a set of increasing values of the scale parameter (called snapshots). We first *independently* strong collapse all these complexes, then assemble the resulting individual *cores* using the induced simplicial maps introduced in Section 4. The resulting new sequence with induced simplicial maps between the collapsed complexes is usually a simplicial tower. We then convert the simplicial tower into an equivalent filtration using the software Sophia [38] that implements an algorithm of Dey et. al. [20]. Finally, we

$\mathcal{X}$	1-sphere	2-Annulus	dragon	netw-sc	senate	eleg
Snp	80	80	46	69	107	77
Flt(10^6)	0.12	13.91	7.96	22.35	2.56	1.18
Twr	54	252	1,641	380	104	298
EqF	573	1,954	8,437	957	270	431
Flt/EqF(10^3)	0.21	7.12	0.94	23.35	9.48	2.74
PDF	0.65	174.18	69.92	243.86	24.92	10.87
MCT	0.005	0.022	0.065	0.009	0.003	0.002
AT	0.045	0.136	0.408	0.078	0.06	0.157
PDT	0.01	0.02	0.08	0.01	0.005	0.006
Total	0.060	0.178	0.553	0.097	0.068	0.165
PDF/Total	10.8	978.5	126.4	2514.0	366.5	65.9

dimension snapshot = all fitration values?

Table 2 The rows are, from top to down: dataset $\mathcal{X}$, number of snapshots (snp), total number of simplices in the original filtration (Flt), number of simplices in the collapsed tower (Twr), total number of simplices in the equivalent filtration (EqF), ratio of Flt and EqF (Flt/EqF), PD computation time for the original filration (PDF), maximum collapse time (MCT), assembly time (AT), PD computation time of the tower (PDT), sum MCT+AT+PDT (Total), ratio of PDF and Total (PDF/Total). All times are noted in seconds. For the first three datasets, we sampled points randomly from the initial datasets and averaged the results over five trials.

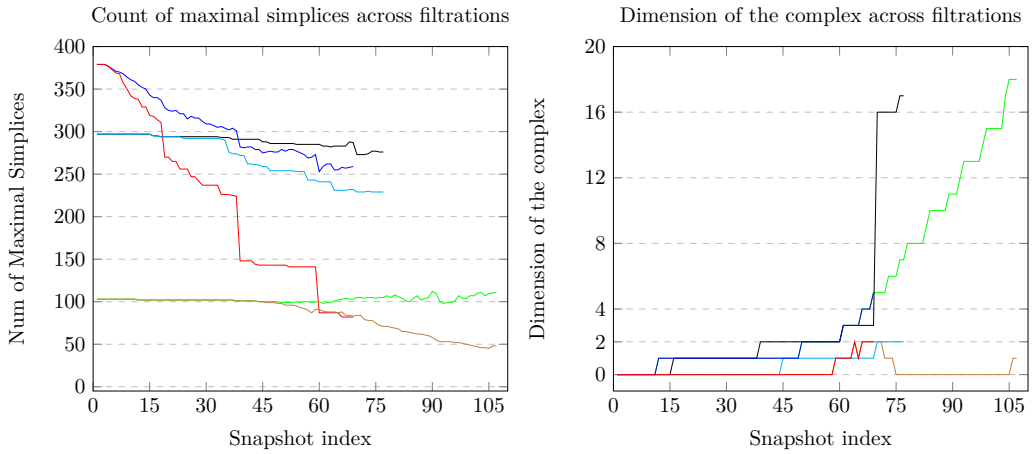
run the persistence algorithm of the Gudhi library [37] to obtain the persistence diagram (PD) of the equivalent filtration. By the results of Section 4, the obtained PD is the same as the PD of the initial sequence.

The total time to compute the PD of the core sequence is the sum of three terms: 1. the *maximum* time taken to collapse all the individual complexes, 2. the time taken to assemble the individual *cores* to form a tower, 3. the time to compute the persistent diagram of the tower. Table 2 summarises the results of the experiments. In both cases, the original filtration and the collapsed tower, we use Gudhi through Sophia which is done through the command <./sophia -cgudhi inputTowerFile outputPDFile>. When we use the -cgudhi option, Sophia reports two computation times, one the total time taken by Sophia that includes (1) reading the tower, (2) transforming it to a filtration and (3) time taken by Gudhi to compute PD, second reported time is just the time taken by Gudhi to compute PD. In our comparison, for the original filtration we report just the Gudhi time and for the collapsed tower we report the total time taken by Sophia.

The dataset of the first column (1-sphere) of Table 2 consists of 100 random points sampled from a unit circle in dimension 2. The dataset of the second column (2-Annulus) consists of 150 random points sampled from a two dimension annulus of radii $\{0.6, 1\}$. For all the other experiments, we use datasets from a publicly available repository [43]. These datasets have been previously used to benchmark different publicly available software computing PH [42]. For the third experiment (*dragon*), we randomly picked 150 points from the 2000 points of the dataset **drag 2** of [43]. The fourth and fifth column respectively correspond to the dataset **netw-sc** and **senate** of [43], here we used the distance matrix. The sixth column corresponds to the dataset **eleg** of [43], and here again we used the distance matrix. The first three datasets are point sets in Euclidean space. For the other three, the distance matrices of the datasets were available at [43]. The [initial value, increment, final value] of the scale

parameter are $[0.1, 0.005, 0.5]$, $[0.1, 0.005, 0.5]$, $[0, 0.001, 0.046]$, $[0.1, 0.05, 3.5]$, $[0, 0.001, 0.107]$ and $[0, 0.001, 0.077]$ for the examples in Table 2 (from left to right). For more detail about the datasets and the computation of the distance matrices of the last three datasets please refer to [42].

The plots below count the maximal simplices and the dimensions of the complexes across the filtration and the collapsed tower. Blue and red correspond respectively to the filtration and the collapsed tower of the data **netw-sc**. Similarly green and brown correspond respectively to the filtration and the collapsed tower of the data **senate**. Finally, **black** and cyan correspond to the filtration and the collapsed tower of the data **eleg** respectively. We can observe that in all cases the number of maximal simplices either stays near constant or decreases. Also they are far fewer in number compared to the total number of simplices. Observe that for the uncollapsed filtrations blue, green and **black**, the dimension of the complexes increases quite rapidly with the snapshot index. Although dimensions are small, their effect on the total number of simplices is exponential. Another key fact to observe is that the dimension of the complexes in the corresponding collapsed tower are much smaller than their counterparts in the filtration.



Noticeably, in our experiments, the computing time of our approach is reduced by 1 to 3 orders of magnitude. The gain is more with the size of the filtration. A similar reduction of 2 to 4 orders of magnitude is achieved for the number of simplices. Observations from the plots combined with the experimental results of Table 2 clearly indicate that our method is extremely fast and memory efficient.

The implementation of the Core algorithm 1 bench-marked here is coded in C++ and will be available as an open-source package of the next release of the Gudhi library [37]. The code was compiled using the compiler <clang-900.0.38> and all computations were performed on <2.8 GHz Intel Core i5> machine with 16 GB of available RAM.

The experiments above are limited to filtrations of VR-complexes, by far the most commonly used type of sequences in Topological Data Analysis. We intend to experiment on Zigzag sequences in future work.

Acknowledgements. We want to thank Marc Glisse for his useful discussions, Mathijs Wintraecken for reviewing a draft of the article. We also want to thank Francois Godi and Siargey Kachanovich for their help with Gudhi and Hannah Schreiber for her help with Sophia.

References

1 J.A. Barmak, E.G. Minian. Strong homotopy types, nerves and collapses. Discrete and Computational Geometry 47(2012), pp. 301-328.

2 M. Tancer: Recognition of collapsible complexes is NP-complete. Discrete and Computational Geometry. 55(1) (2016), 21-38.

3 D. Attali, A. Lieutier, and D. Salinas. Efficient data structure for representing and simplifying simplicial complexes in high dimensions. International Journal of Computational Geometry and Applications (IJCGA), 22(4):279–303, 2012.

4 A. C. Wilkerson, T. J. Moore, A. Swami,and A. H. Krim. Simplifying the homology of networks via strong collapses. ICASSP - Proceedings (2013).

5 A. C. Wilkerson, H. Chintakunta and H. Krim. Computing persistent features in big data: A distributed dimension reduction approach. ICASSP - Proceedings (2014) pp. 11-15..

6 C. H. Dowker, "Homology groups of relations," The Annals of Mathematics, vol. 56, pp. 84–95, 1952.

7 M. Adamaszek and J. Stacho. Complexity of simplicial homology and independence complexes of chordal graphs. Computational Geometry: Theory and Applications. 57, 8–18 (2016).

8 K. Mischaikow and V. Nanda. "Morse Theory for Filtrations and Efficient Computation of Persistent Homology". DCG, Volume 50, Issue 2, pp 330-353, September 2013.

9 Michael Kerber and R. Sharathkumar. "Approximate Čech Complex in Low and High Dimensions". In: Algorithms and Computation. Ed. by Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam. Vol. 8283. Lecture Notes in Computer Science. 2013, pp. 666–676.

10 Donald Sheehy."Linear-Size Approximations to the Vietoris–Rips Filtration". In: Discrete and Computational Geometry 49.4 (2013), pp.778–796.

11 Afra Zomorodian. "The Tidy Set: A Minimal Simplicial Set for Computing Homology of Clique Complexes". In: Proceedings of the Twenty-sixth Annual Symposium on Computational Geometry. SoCG'10. Snowbird, Utah, USA, 2010, pp. 257–266. isbn: 978-1-4503-0016-2.

12 M. Botnan. and G Spreemann. "Approximating persistent homology in Euclidean space through collapses". In: Applicable Algebra in Engineering, Communication and Computing 26 (1-2), 73-101.

13 P. Dłotko, H. Wagner, Simplification of complexes for persistent homology computations, Homology, Homotopy and Applications, vol. 16(1), 2014, pp.49 - 63.

14 T. K. Dey, H. Edelsbrunner, S. Guha and D. Nekhayev. Topology preserving edge contraction. . Publications de l'Institut Mathematique (Beograd), Vol. 60 (80), 23–45, 1999.

15 C. Chen and M. Kerber. Persistent homology computation with a twist. In European Workshop on Computational Geometry (EuroCG), pages 197-200, 2011.

16 U. Bauer, M. Kerber, and J. Reininghaus. Clear and compress: Computing persistent homology in chunks. In Topological Methods in Data Analysis and Visualization III, Mathematics and Visualization, pages 103-117. Springer, 2014.

17 François Le Gall. Powers of tensors and fast matrix multiplication. ISSAC '14, pages 296–303. ACM, 2014.

18 A. Zomorodian and G. Carlsson. Computing persistent homology. Discrete Comput. Geom., 33(2):249–274, 2005.

19 H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. Discrete Comput. Geom., 28:511–533, 2002.

20 T. Dey, F. Fan, and Y. Wang. Computing Topological Persistence for Simplicial Maps. In ACM Symposium on Computational Geometry (SoCG), pages 345-354, 2014.

21 F. Chazal, S. Oudot. Towards Persistence-Based Reconstruction in Euclidean Spaces. In Proc. ACM Symp. of Computational Geometry 2008.

22 Gunnar Carlsson and Vin de Silva. "Zigzag Persistence". Found Comput Math (2010) 10: 367.
23 Gunnar Carlsson, Vin de Silva, and Dmitriy Morozov. Zigzag persistent homology and real-valued functions. SOCG (2009), pages 247-256.
24 C. Maria and S. Oudot. "Zigzag Persistence via Reflections and Transpositions". In: Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA). Jan. 2015, pp. 181–199.
25 Nikola Milosavljevic, Dmitriy Morozov, and Primoz Skraba. Zigzag persistent homology in matrix multiplication time. In Symposium on Comp. Geom., 2011.
26 Harm Derksen and Jerzy Weyman. Quiver representations. Notices of the American Mathematical Society, 52(2):200–206, February 2005.
27 Cohen-Steiner, David; Edelsbrunner, Herbert and Harer,John. Stability of Persistence Diagrams. Discrete Comput. Geom., 37(1):103-120, 2007.
28 Herbert Edelsbrunner and John Harer. Computational Topology: An Introduction. American Mathematical Society, 2010.
29 A. Hatcher. Algebraic Topology. Cambridge Univ. Press, 2001.
30 J.H.C Whitehead. Simplicial spaces, nuclei and m-groups. Proc. London Math. Soc. 45 (1939), 243-327.
31 Jean-Daniel Boissonnat, Karthik C. S., Sébastien Tavenas: Building Efficient and Compact Data Structures for Simplicial Complexes. Algorithmica 79(2): 530-567 (2017).
32 Jean-Daniel Boissonnat, Karthik C. S.: An Efficient Representation for Filtrations of Simplicial Complexes. ACM-SIAM Symposium on Discrete Algorithms (SODA 2017).
33 Gunnar Carlsson, Tigran Ishkhanov, Vin de Silva, and Afra Zomorodian. "On the Local Behavior of Spaces of Natural Images". In: International Journal of Computer Vision 76.1 (2008), pp. 1–12.
34 Joseph Minhow Chan, Gunnar Carlsson, and Raul Rabadan. "Topology of viral evolution". In: Proceedings of the National Academy of Sciences 110.46 (2013), pp. 18566–18571.
35 Vin de Silva and Robert Ghrist. "Coverage in sensor networks via persistent homology". In: Algebraic and Geometric Topology 7 (2007), pp. 339 – 358.
36 Jose Perea and Gunnar Carlsson. "A Klein-Bottle-Based Dictionary for Texture Representation". In: International Journal of Computer Vision 107.1 (2014), pp. 75–97.
37 Gudhi: Geometry understanding in Higher Dimensions. http://gudhi.gforge.inria.fr/
38 Sophia: https://bitbucket.org/schreiberh/sophia/
39 Phat: https://bitbucket.org/phat-code/phat
40 Ripser: https://github.com/Ripser/ripser
41 Dionysus: http://www.mrzv.org/software/dionysus/
42 N. Otter, M. Porter, U. Tillmann, P. Grindrod and H. Harrington. "A roadmap for the computation of persistent homology", EPJ Data Science 2017 6:17, Springer Nature.
43 Datasets: "https://github.com/n-otter/PH-roadmap/"

6 Appendix

Proof of Theorem 1 : We recall the following crucial lemma from [1, Lemma 3.3], required to prove the Theorem 1 .

▶ **Lemma 3.** *Let L be a full subcomplex of a simplicial complex K, such that every vertex $v \in K, \notin L$ is dominated by some vertex in L, then $K \searrow\searrow L$.*

The vertices of $\mathcal{N}^2(K)$ are the maximal family of the maximal simplices of K that has a common intersection. That is, if $\Sigma_s = \{\sigma_0^s, \sigma_1^s, ..., \sigma_{i_s}^s\}$ is a vertex of $\mathcal{N}^2(K)$, then is the

maximal set such that $\bigcap_{0\leq j\leq i_s} \sigma_j^s \neq \emptyset$, where the $\sigma_j^s s$ are maximal simplices of K. A $1-$simplex of $\mathcal{N}^2(K)$ is a pair Σ_s and Σ_t such that $\Sigma_s \cap \Sigma_t \neq \emptyset$ and $\Sigma_s \cap \Sigma_t$ is a set of maximal simplices of K. Otherwise $\Sigma_s \cup \Sigma_t$ would be a maximal family of intersecting maximal simplices, contradicting the maximality of Σ_s and Σ_t. Similarly, all the higher dimensional simplices will follows this condition of existence.

Now we define a map $\phi : \mathcal{N}^2(K) \to K$, such that, if Σ_s is a vertex of $\mathcal{N}^2(K)$ then $\phi(\Sigma_s) \in \bigcap_{0\leq j\leq i_s} \sigma_j^s$. We claim that ϕ is injective, simplicial and its image is a full subcomplex of K.

ϕ is injective: Let Σ_0 and Σ_1 be two vertices of $\mathcal{N}^2(K)$, such that $\phi(\Sigma_0) = \phi(\Sigma_1) = v \in K$. Then $v \in \bigcap_{0\leq j\leq i_0} \sigma_j^0$ and $v \in \bigcap_{0\leq j\leq i_1} \sigma_j^1$, which implies v is a common vertex in the maximal simplices which defines both Σ_0 and Σ_1. Therefore $\Sigma_0 \cup \Sigma_1$ is a family of interesecting maximal simplices of K. Since Σ_0 and Σ_1 are maximal, $\Sigma_0 = \Sigma_1 = \Sigma_0 \cup \Sigma_1$.

ϕ is simplicial: Let $[\Sigma_0, \Sigma_1, ..., \Sigma_k]$ be a k-simplex of $\mathcal{N}^2(K)$, then there exists a maximal simplex σ of K, such that, $\sigma \in \Sigma_s, \forall s = 0...k$. By definition of ϕ, $\phi(\Sigma_s) \in \sigma$ for all s. Therefore $\phi([\Sigma_0, \Sigma_1, ..., \Sigma_k])$ spans a face of σ, which is a simplex in K.

$\phi(\mathcal{N}^2(K))$ is a full subcomplex of K : Let σ be a simplex of K spanned by $\{\phi(\Sigma_0), \phi(\Sigma_1), ..., \phi(\Sigma_r)\}$ and let $\sigma^{'}$ is a maximal simplex of K, such that $\sigma \subseteq \sigma^{'}$. Since $\phi(\Sigma_s) \in \sigma \subseteq \sigma^{'}$, $\sigma' \in \Sigma_s, \forall s = 0...r$, hence $\{\Sigma_0, \Sigma_1, ..., \Sigma_r\}$ spans a simplex in $\mathcal{N}^2(K)$.

The existence of such a map ϕ proves the following lemma.

▶ **Lemma 4.** *$\mathcal{N}^2(K)$ is isomorphic to a full subcomplex of K.*

Let v be a vertex of K, which is not in the image $\phi(\mathcal{N}^2(K))$. Let $\Sigma = \{\sigma_0, \sigma_1, ..., \sigma_l\}$ be the set of all the maximal simplices of K that contains v. Since v is not in the image of ϕ, Σ may or may not be maximal. Hence there exists a vertex Σ_s of $\mathcal{N}^2(K)$, such that, $\Sigma \subseteq \Sigma_s$. By definition $\phi(\Sigma_s)$ dominates v. Therefore $\phi(\mathcal{N}^2(K))$ satisfies the hypothesis of Lemma 6, which implies Theorem 1.